Implementing ACPI 5 Features
Device Enumeration And Power Management

Rafael J. Wysocki

Intel Open Source Technology Center

April 16, 2013
1. Introduction: ACPI 5
   - ACPI Overview
   - What’s New In ACPI 5

2. ACPI GPIO And SPB Support In The Kernel
   - Representation Of Controllers
   - Enumeration Of Client Devices

3. ACPI Device Power Management
   - ACPI PM Control Interface
   - ACPI PM Domain

4. Resources
General Information About ACPI

Advanced Configuration and Power Interface Specification
Rules for communication between the platform firmware and the OS.

Current revision: ACPI 5.0
Published on December 6, 2011 (http://www.acpi.info).

ACPI Namespace
Hierarchical structure containing data and code.
- ACPI Machine Language (AML)
- ACPI Source Language (ASL)
The ACPI CA Project

ACPI Component Architecture

OS-independent reference implementation of the ACPI specification.

1. Common in-kernel code (shared by multiple OSes including Linux*).
   - AML execution (AML interpreter)
   - Parsing of ACPI tables
   - Handling of events

2. AML compiler and disassembler.

3. ACPI-related utilities.

Linux kernel’s ACPICA code in sync with upstream (Intel OTC).

More information: https://www.acpica.org
Major ACPI 5 Changes And New Features

- Hardware-reduced ACPI model.
- Memory power management support.
- Generic interrupt controller support.
- Fixed DMA descriptor support.
- GPIO abstraction.
- Simple peripheral bus (I²C, SPI, UART) abstraction.
- Platform Communication Channel (PCC).
- Collaborative Processor Performance Control (CPPC).
- New time and alarm device abstraction.

More information: Len Brown’s presentation *ACPI5 on Linux* [3].
GPIO And Simple Peripheral Bus (SPB) Support

Connection Resources

1. GPIO and SPB controllers as devices in the ACPI Namespace.
2. Connections to those controllers abstracted as hardware resources.
   - New resource types (GPIO, I^2C, SPI, UART connections).
   - _CRS, _SRS, _PRS for slave (client) device objects.
3. Operation Regions referring to connection resources.

Mapping of GPIO interrupts to ACPI events

1. Interrupts listed by _AEI.
2. Handled through _Exx, _Lxx, or _EVT.
How To Represent GPIO And SPB Controllers

Depending on what bus they are

- PCI : struct pci_dev
- Unspecified : struct platform_device

Rationale

Existing platform drivers for functionally identical hardware.

Goal

Make it as simple as possible to re-use drivers.

Problem with the legacy

List of IDs to represent as struct platform_device in ACPI core.
#ifdef CONFIG_ACPI
...

static struct acpi_device_id pxa2xx_spi_acpi_match[] = {
    { "INT33C0", 0 },
    { "INT33C1", 0 },
    { },
};
MODULE_DEVICE_TABLE(acpi, pxa2xx_spi_acpi_match);
#endif

static struct platform_driver driver = {
    ...
    .driver = {
        .acpi_match_table = ACPI_PTR(pxa2xx_spi_acpi_match),
        ...
    },
};
SPI Serial Bus

`acpi_register_spi_devices()`

Run by the SPI core subsystem.
#ifdef CONFIG_ACPI
static struct acpi_device_id at25_acpi_match[] = {
    { "AT25", 0 },
    { },
};
MODULE_DEVICE_TABLE(acpi, at25_acpi_match);
#endif

static struct spi_driver at25_driver = {
    .driver = {
        ...
        .acpi_match_table = ACPI_PTR(at25_acpi_match),
    },
};
acpi_i2c_register_devices()

Run by the controller driver.
GPIO Pin Lookup

ACPI Namespace

- PNPID
- _CRS
- GpioIo
- INT33C7

Linux device hierarchy

- struct platform_device
- GPIO number
- struct gpio_chip

acpi_get_gpio()
Device Power States And PM Methods

Device power states

\[ D_0, D_1, D_2, D_{3_{hot}}, D_{3_{cold}} \]

Device-specific power control methods

\_PS0, \_PS1, \_PS2, \_PS3, \_PSC, \_PSE

Power resources interface

Power resources : \_ON, \_OFF, \_STA

Device objects : \_PR0, \_PR1, \_PR2, \_PR3, \_PRE

Wakeup interface

\_DSW, \_PRW
Device PM Rules – acpi_device_set_power()

To put device into $D_0$

1. All _PR0 power resources must be ON.
2. Execute _PS0.

Analogously for $D_1$, $D_2$, $D_{3_{hot}}$

Transitions from $D_{3_{hot}}$/$D_{3_{cold}}$ only to $D_0$.

To put device into $D_{3_{cold}}$

1. Put it into $D_{3_{hot}}$.
2. Turn off all _PR0, _PR1, _PR2, _PR3 power resources.
General PM Domain For ACPI PM

Goal
Use ACPI device PM in the existing PM frameworks.

Observation
No native PM for platform/SPI/I²C bus types.

Idea
ACPI PM domain providing subsystem-level device PM callbacks.
- `acpi_dev_pm_attach()`
- `acpi_dev_pm_detach()`
ACPI Device PM Code Flow (Runtime PM)

Runtime Suspend

1. PM Core
2. ACPI PM Domain
3. Driver .runtime_suspend()
4. acpi_dev_runtime_suspend()

Runtime Resume

1. Driver .runtime_resume()
2. acpi_dev_runtime_resume()
3. ACPI PM Domain
4. PM Core
Summary

- More configuration information in the ACPI namespace with ACPI 5.
- Devices using SPB and GPIO connections identifiable.
- SPB/GPIO controllers with no specific bus type represented as platform devices.
- Straightforward driver re-use for devices enumerated through ACPI.
- ACPI PM interface applicable more widely.
References

*Advanced Configuration and Power Interface Specification, Revision 5.0*

L. Duflot, O. Levillain, B. Morin, *ACPI Design Principles and Concerns*

Len Brown, *ACPI5 on Linux*
Documentation And Source Code

- https://www.acpica.org
- Documentation/acpi/enumeration.txt
- include/acpi/acpi_bus.h
- include/linux/acpi.h
- include/linux/device.h
- drivers/acpi/acpi_platform.c
- drivers/acpi/acpi_i2c.c
- drivers/gpio/gpiolib-acpi.c
- drivers/spi/spi.c
- drivers/i2c/busses/i2c-designware-platdrv.c
- drivers/spi/spi-pxa2xx.c
- drivers/mmc/host/sdhci-acpi.c
Thanks!

Thank you for attention!